

计算机学院专业必修课

---

# 计算机组成

## 从晶体管到运算

高小鹏

北京航空航天大学计算机学院  
系统结构研究所

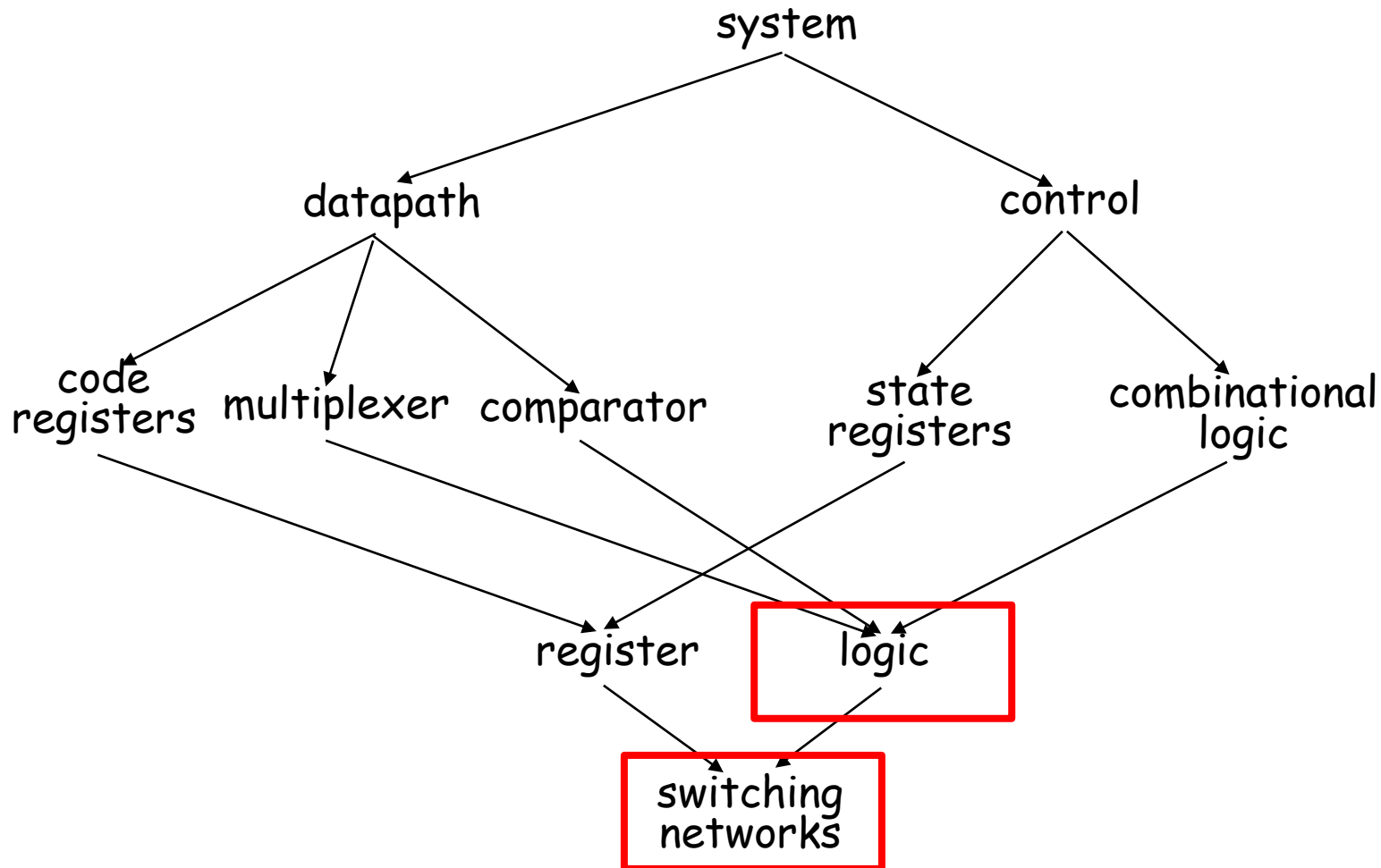
# 提纲

- 内容主要取材：CS61C的17讲、18讲
  - <http://inst.eecs.berkeley.edu/~cs61c/su12>
- 晶体管
- 门电路
- 运算

# 提纲

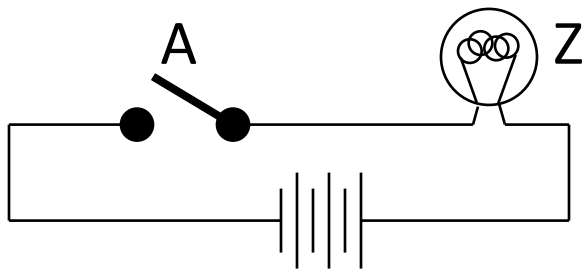
- 内容主要取材：CS61C的17讲、18讲
  - <http://inst.eecs.berkeley.edu/~cs61c/su12>
- 晶体管
- 门电路
- 运算

# Design Hierarchy

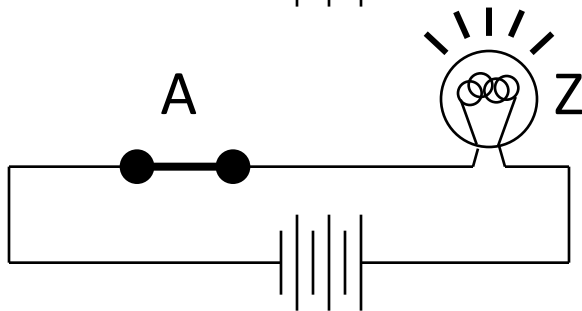


# Switches (1/2)

- The basic element of physical implementations
- Convention: if input is a “1,” the switch is *asserted*



*Open* switch if A is “0” (unasserted)  
and turn OFF light bulb (Z)

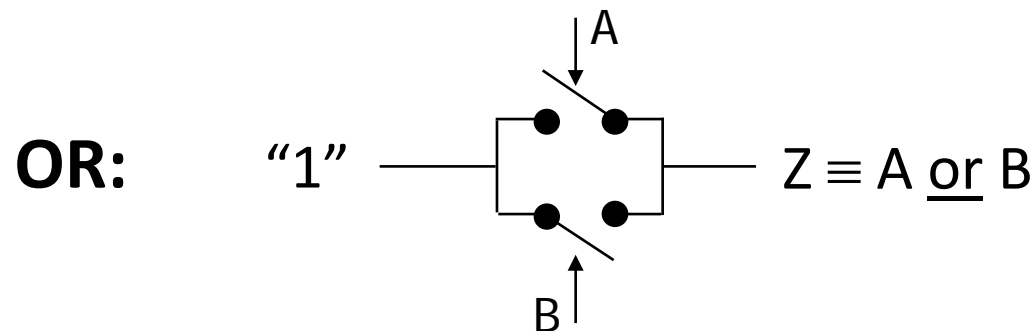
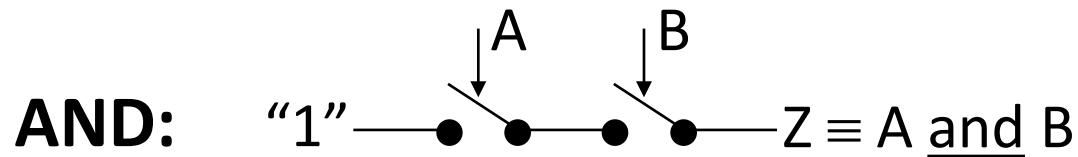


*Close* switch if A is “1” (asserted)  
and turn ON light bulb (Z)

In this example,  $Z \equiv A$ .

## Switches (2/2)

- Can compose switches into more complex ones (Boolean functions)
  - Arrows show action upon assertion (1 = close)



# Transistors

- High voltage ( $V_{dd}$ ) represents 1, or true
- Low voltage (0 volts or Ground) represents 0, or false
- Let threshold voltage ( $V_{th}$ ) decide if a 0 or a 1
  - $V_{th}$ 一般小于 $V_{dd}$
- If switches control whether voltages can propagate through a circuit, can build a computer
- Our switches: CMOS transistors

# CMOS Transistor Networks

- Modern digital systems designed in CMOS
  - MOS: Metal-Oxide on Semiconductor
  - C for complementary: use *pairs* of normally-open and normally-closed switches
    - Used to be called COS-MOS for complementary-symmetry - MOS
- CMOS transistors act as voltage-controlled switches
  - Similar, though easier to work with, than relay switches from earlier era (Porter computer)
  - Use energy primarily when switching

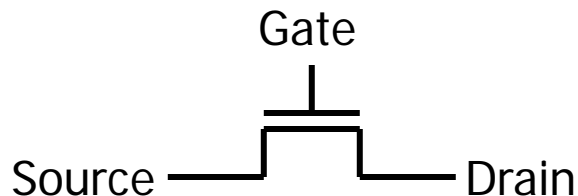
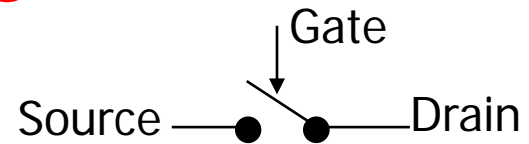


# CMOS Transistors

- Three terminals: source, gate, and drain

- Switch action:

if voltage on gate terminal is (some amount) higher/lower than source terminal then conducting path established between drain and source terminals (switch is closed)



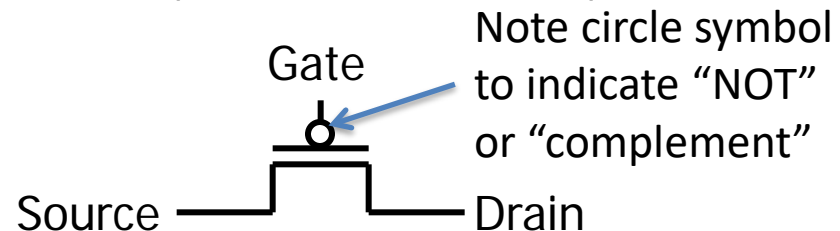
*n-channel transistor*

**open** when voltage at Gate is low

**closes** when:

voltage(Gate) > voltage (Threshold)

(High resistance when gate voltage Low,  
Low resistance when gate voltage High)



*p-channel transistor*

**closed** when voltage at Gate is low

**opens** when:

voltage(Gate) > voltage (Threshold)

(Low resistance when gate voltage Low,  
High resistance when gate voltage High)

# CMOS circuit rules

要点1:  
导通时应该让  
source电压与  
gate电压有反差

- Don't pass weak values => Use Complementary Pairs
  - ✗ – N-type transistors pass weak 1's ( $V_{dd} - V_{th}$ )
  - ✓ – N-type transistors pass strong 0's (ground)
    - Use N-type transistors only to pass 0's (N for negative)
    - Converse for P-type transistors: Pass weak 0s, strong 1s
      - Pass weak 0's ( $V_{th}$ ), strong 1's ( $V_{dd}$ )
      - Use P-type transistors only to pass 1's (P for positive)
    - Use pairs of N-type and P-type to get strong values
- Never leave a wire undriven
  - Make sure there's always a path to  $V_{dd}$  or gnd
- Never create a path from  $V_{dd}$  to gnd (ground)

要点2:  
避免无源驱动  
要点3:  
避免短路

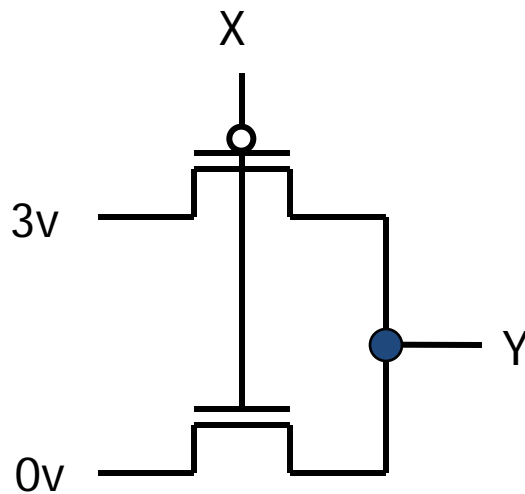
# MOS Networks

*p-channel transistor*

closed when voltage at Gate is low

opens when:

$\text{voltage}(\text{Gate}) > \text{voltage}(\text{Threshold})$



*n-channel transistor*

open when voltage at Gate is low

closes when:

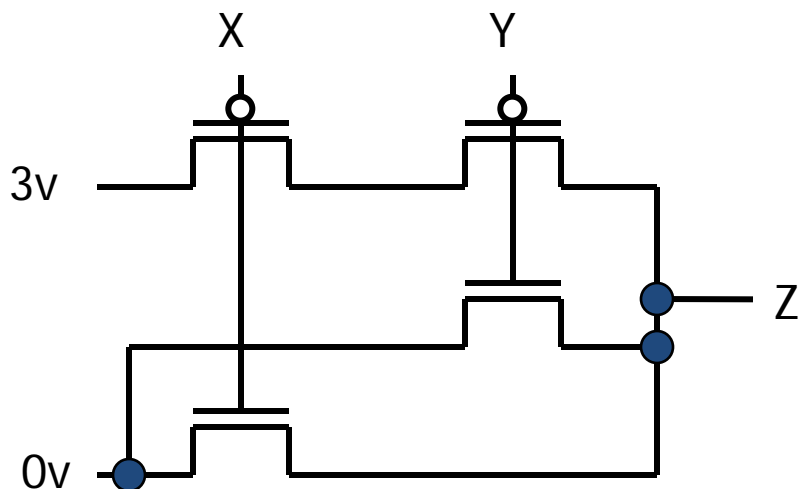
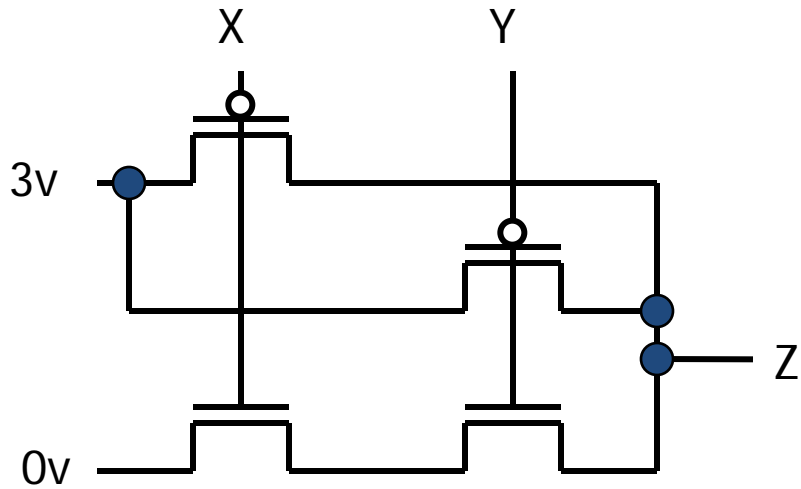
$\text{voltage}(\text{Gate}) > \text{voltage}(\text{Threshold})$

what is the  
relationship  
between x and y?

x	y
0 volts (gnd)	3 volts (Vdd)
3 volts (Vdd)	0 volts (gnd)

Called an *inverter* or *not gate*

# Two Input Networks

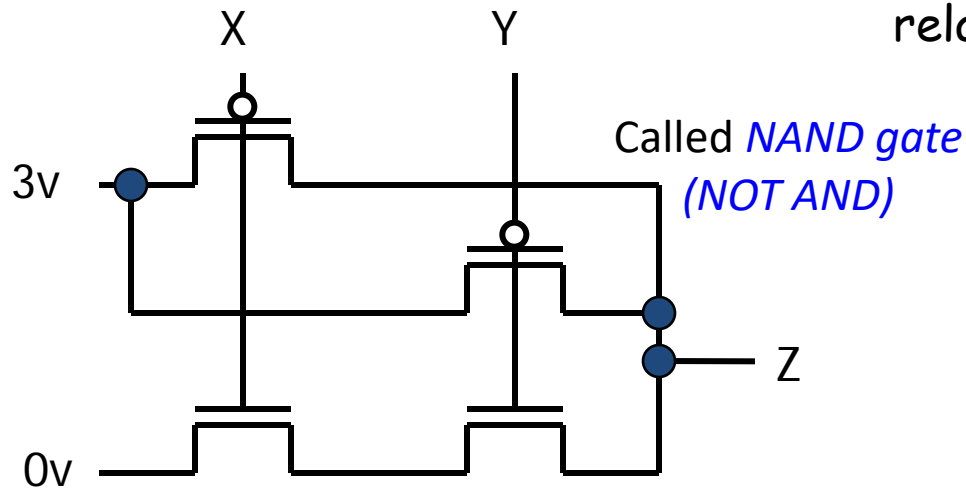


what is the  
relationship between x, y and z?

x	y	z
0 volts	0 volts	
0 volts	3 volts	
3 volts	0 volts	
3 volts	3 volts	

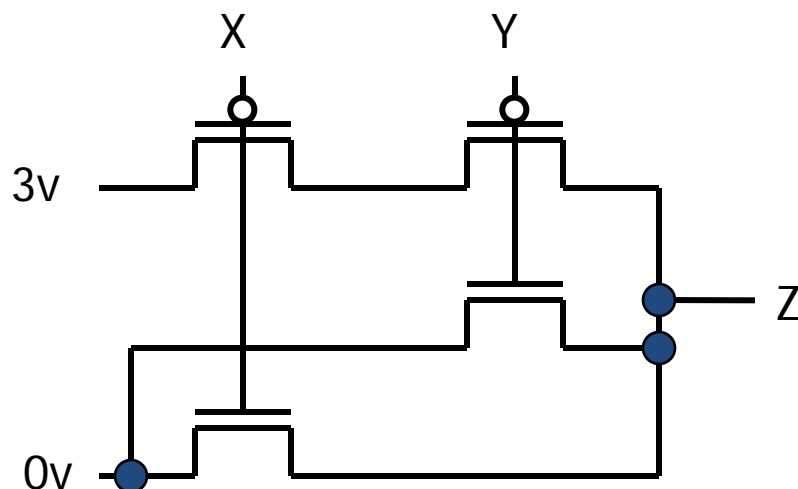
x	y	z
0 volts	0 volts	
0 volts	3 volts	
3 volts	0 volts	
3 volts	3 volts	

# Two Input Networks: Peer Instruction



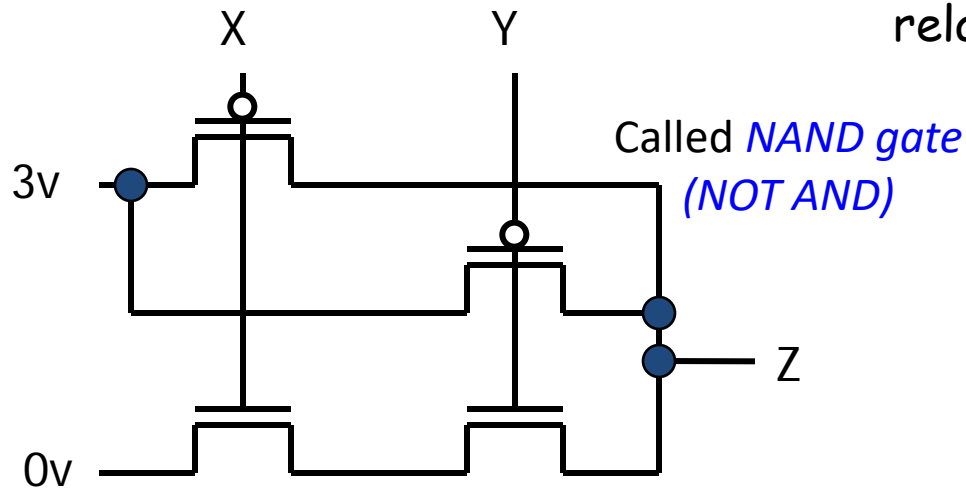
what is the  
relationship between x, y and z?

x	y	z
0 volts	0 volts	3 volts
0 volts	3 volts	3 volts
3 volts	0 volts	3 volts
3 volts	3 volts	0 volts



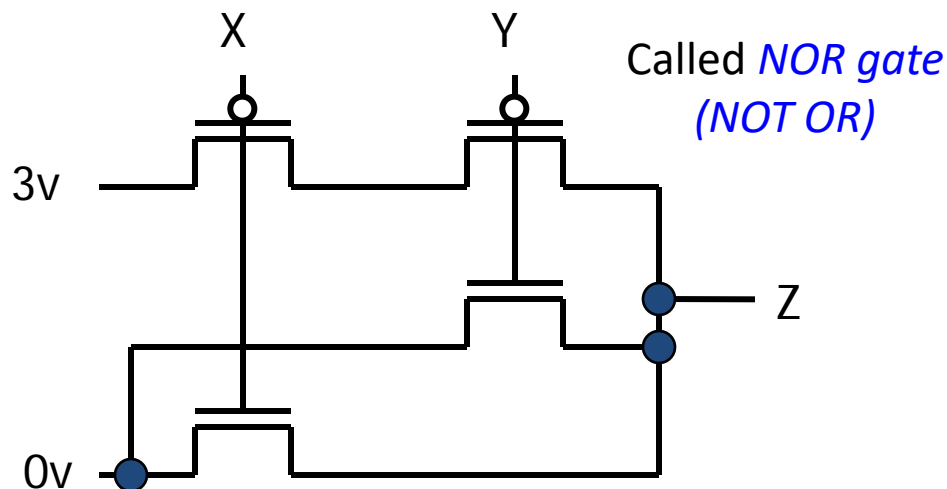
x	y	z			
		A	B	C	D
0 volts	0 volts	0	0	3	3
0 volts	3 volts	0	3	0	3
3 volts	0 volts	0	3	0	3
3 volts	3 volts	3	3	0	0

# Two Input Networks



what is the  
relationship between x, y and z?

x	y	z
0 volts	0 volts	3 volts
0 volts	3 volts	3 volts
3 volts	0 volts	3 volts
3 volts	3 volts	0 volts



x	y	z
0 volts	0 volts	3 volts
0 volts	3 volts	0 volts
3 volts	0 volts	0 volts
3 volts	3 volts	0 volts

# 提纲

- 内容主要取材：CS61C的17讲、18讲
  - <http://inst.eecs.berkeley.edu/~cs61c/su12>
- 晶体管
- 门电路
- 运算

# Type of Circuits

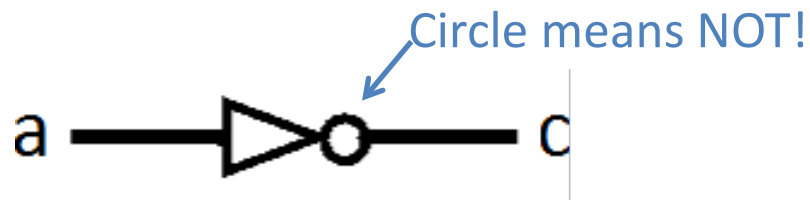
- *Synchronous Digital Systems* consist of two basic types of circuits:
  - Combinational Logic (CL)
    - Output is a function of the inputs only, not the history of its execution
    - e.g. circuits to add A, B (ALUs)
  - Sequential Logic (SL)
    - Circuits that “remember” or store information
    - a.k.a. “State Elements”
    - e.g. memory and registers (Registers)



# Logic Gates (1/2)

- Special names and symbols:

**NOT**



$a$	$c$
0	1
1	0

**AND**



$a$	$b$	$c$
0	0	0
0	1	0
1	0	0
1	1	1

**OR**



$a$	$b$	$c$
0	0	0
0	1	1
1	0	1
1	1	1

# Logic Gates (2/2)

- Special names and symbols:

**NAND**



**NOR**



**XOR**



a	b	c
0	0	1
0	1	1
1	0	1
1	1	0

a	b	c
0	0	1
0	1	0
1	0	0
1	1	0

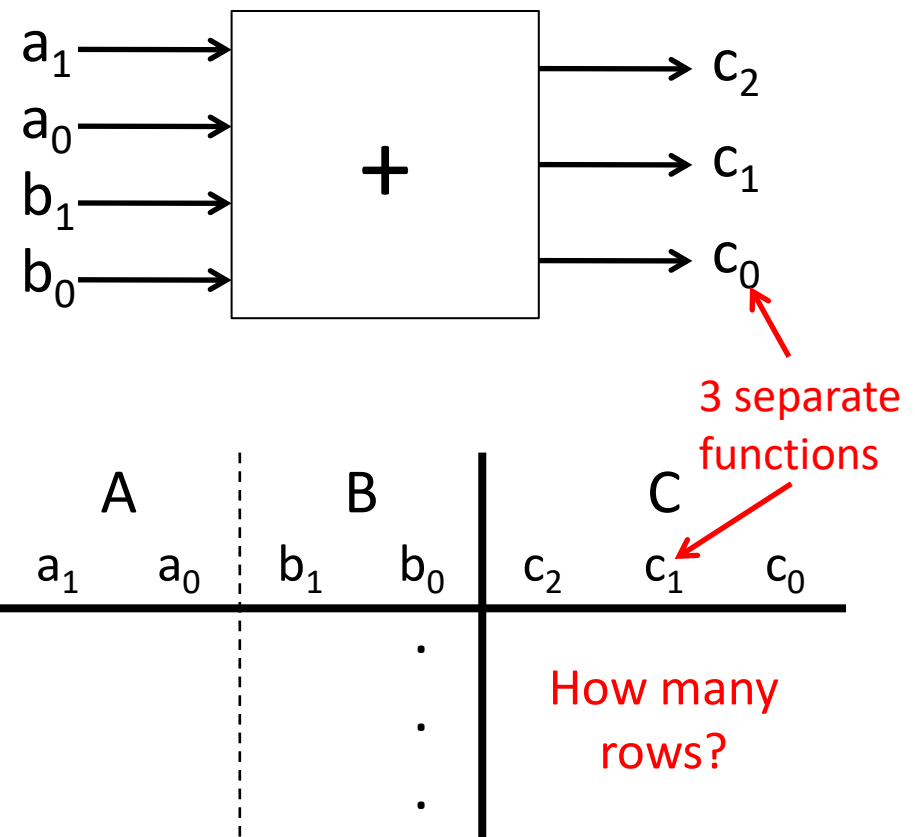
a	b	c
0	0	0
0	1	1
1	0	1
1	1	0

# More Complicated Truth Tables

## 3-Input Majority

a	b	c	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

## 2-bit Adder



# Truth Table to Boolean Expression

- Read off of table

- For 1, write variable name
- For 0, write complement of variable

- *Sum of Products (SoP)*

- Take rows with 1's in output column, sum products of inputs
- $c = \bar{a}b + \bar{b}a$

- *Product of Sums (PoS)*

- Take rows with 0's in output column, product the sum of the *complements of the inputs*
- $c = (a + b) \cdot (\bar{a} + \bar{b})$

a	b	c
0	0	0
0	1	1
1	0	1
1	1	0

# 1.3等值演算

- 定义1.9 设A,B是公式，如果对于每个真值赋值 $v$ ， $v(A)=v(B)$ ，则称A和B等值，也称A与B逻辑等价，记为 $A \Leftrightarrow B$ 。
- 判断 $\neg(p \vee q)$ 和 $\neg p \wedge \neg q$ 是否等值。

<b>p</b>	<b>q</b>	$p \vee q$	$\neg(p \vee q)$	$\neg p$	$\neg q$	$\neg p \wedge \neg q$
<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>
<b>0</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>
<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>
<b>1</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>



# 等值式模式

## ■ 零律

- $A \vee 1 \Leftrightarrow 1 \quad A \wedge 0 \Leftrightarrow 0$

## ■ 幂等律

- $A \vee A \Leftrightarrow A$   
 $A \wedge A \Leftrightarrow A$

## ■ 吸收律

- $A \vee (A \wedge B) \Leftrightarrow A$   
 $A \wedge (A \vee B) \Leftrightarrow A$

## ■ 同一律

- $A \wedge 1 \Leftrightarrow A$
- $A \vee 0 \Leftrightarrow A$
- $A \oplus 0 \Leftrightarrow A$

## ■ 双重否定

- $\neg\neg A \Leftrightarrow A$

## ■ 矛盾律

- $A \wedge \neg A \Leftrightarrow 0$

## ■ 排中律

- $A \vee \neg A \Leftrightarrow 1$

## ■ 假言易位

- $A \rightarrow B \Leftrightarrow \neg B \rightarrow \neg A$



# 等值式模式

## ■ 德·摩根律

- $\neg(A \vee B) \Leftrightarrow \neg A \wedge \neg B$

- $\neg(A \wedge B) \Leftrightarrow \neg A \vee \neg B$

## ■ 交换律

- $A \vee B \Leftrightarrow B \vee A$

- $A \wedge B \Leftrightarrow B \wedge A$

- $A \oplus B \Leftrightarrow B \oplus A$



# 等值式模式

## ■ 结合律

- $(A \vee B) \vee C \Leftrightarrow A \vee (B \vee C)$
- $(A \wedge B) \wedge C \Leftrightarrow A \wedge (B \wedge C)$
- $(A \oplus B) \oplus C \Leftrightarrow A \oplus (B \oplus C)$

## ■ 分配律

- $A \vee (B \wedge C) \Leftrightarrow (A \vee B) \wedge (A \vee C)$
- $A \wedge (B \vee C) \Leftrightarrow (A \wedge B) \vee (A \wedge C)$
- $A \wedge (B \oplus C) \Leftrightarrow (A \wedge B) \oplus (A \wedge C)$





# 等值式模式

- $A \oplus A \Leftrightarrow 0$
- $A \oplus 1 \Leftrightarrow \neg A$
- $A \rightarrow B \Leftrightarrow \neg A \vee B$
- $A \leftrightarrow B \Leftrightarrow (A \rightarrow B) \wedge (B \rightarrow A)$
- $A \oplus B \Leftrightarrow (\neg A \wedge B) \vee (A \wedge \neg B)$
- $A \oplus B \Leftrightarrow \neg(A \leftrightarrow B)$



# Laws of Boolean Algebra

These laws allow us to perform simplification:

$$x \cdot \bar{x} = 0$$

$$x \cdot 0 = 0$$

$$x \cdot 1 = x$$

$$x \cdot x = x$$

$$x \cdot y = y \cdot x$$

$$(xy)z = x(yz)$$

$$x(y + z) = xy + xz$$

$$xy + x = x$$

$$\bar{x}y + x = x + y$$

$$\overline{x \cdot y} = \bar{x} + \bar{y}$$

$$x + \bar{x} = 1$$

$$x + 1 = 1$$

$$x + 0 = x$$

$$x + x = x$$

$$x + y = y + x$$

$$(x + y) + z = x + (y + z)$$

$$x + yz = (x + y)(x + z)$$

$$(x + y)x = x$$

$$(\bar{x} + y)x = xy$$

$$\overline{x + y} = \bar{x} \cdot \bar{y}$$

complementarity

laws of 0's and 1's

identities

idempotent law

commutativity

associativity

distribution

uniting theorem

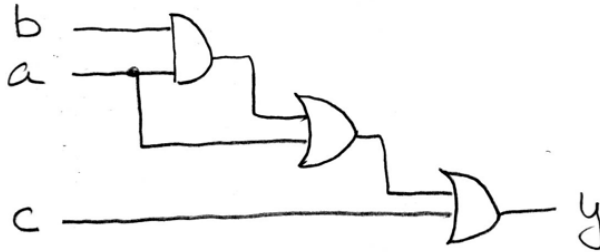
uniting theorem v.2

DeMorgan's Law

# Boolean Algebraic Simplification Example

$$y = ab + a + c$$

# Circuit Simplification



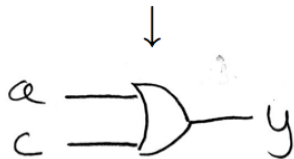
1) original circuit (Transistors and/or Gates)

$$y = ((ab) + a) + c$$

2) equation derived from original circuit

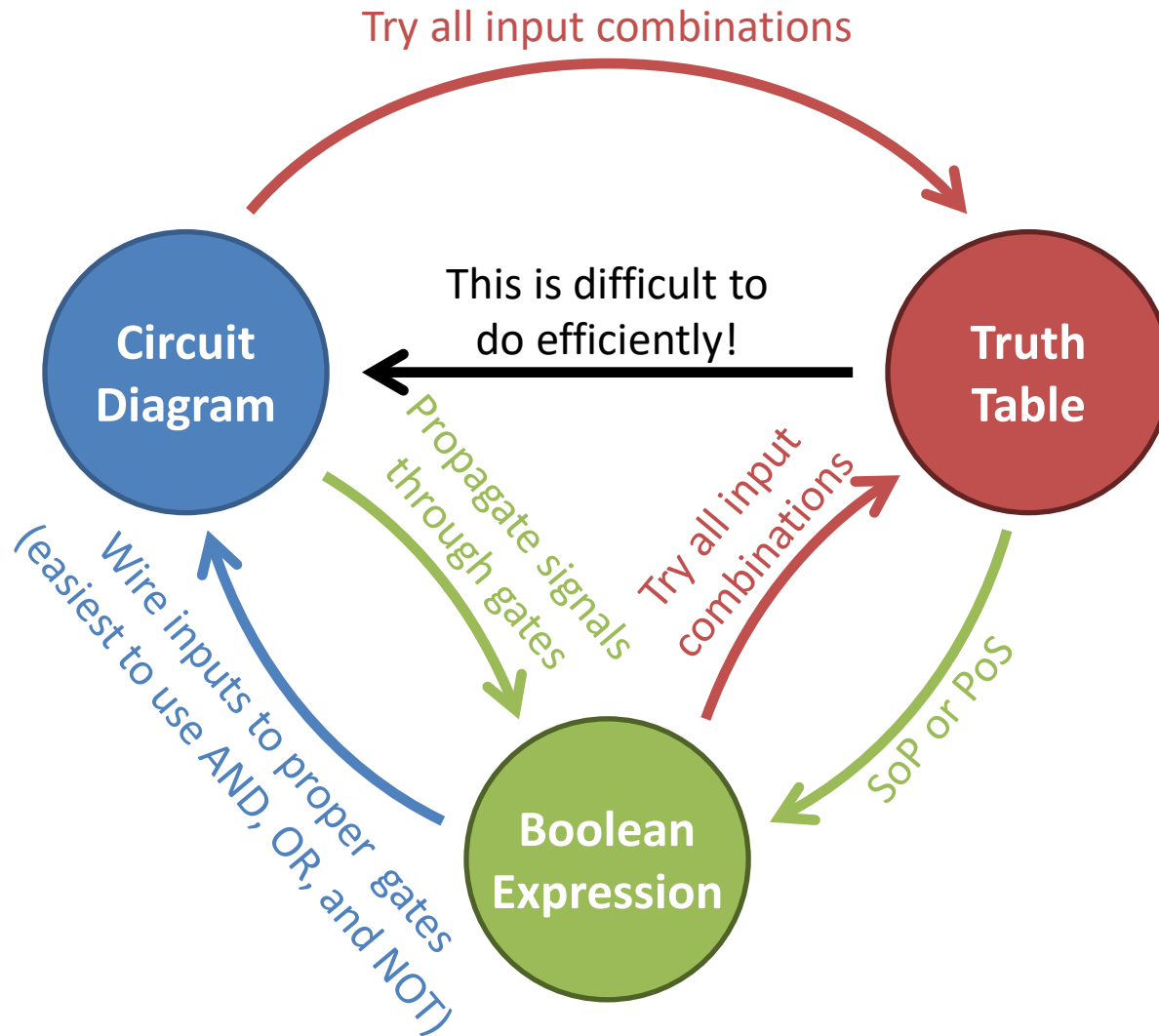
$$\begin{aligned} &\downarrow \\ &= ab + a + c \\ &\downarrow \\ &= a(b + 1) + c \\ &= a(1) + c \\ &= a + c \end{aligned}$$

3) algebraic simplification



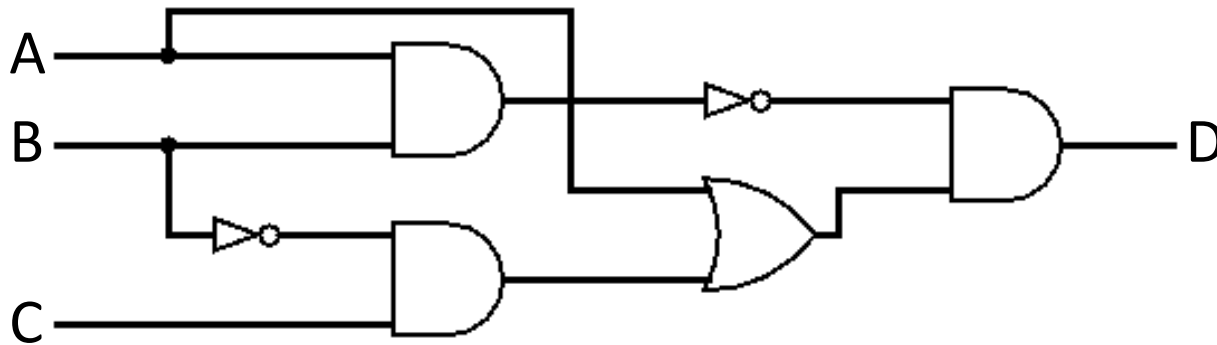
4) simplified circuit

# Converting Combinational Logic



# Circuit Simplification Example (1/4)

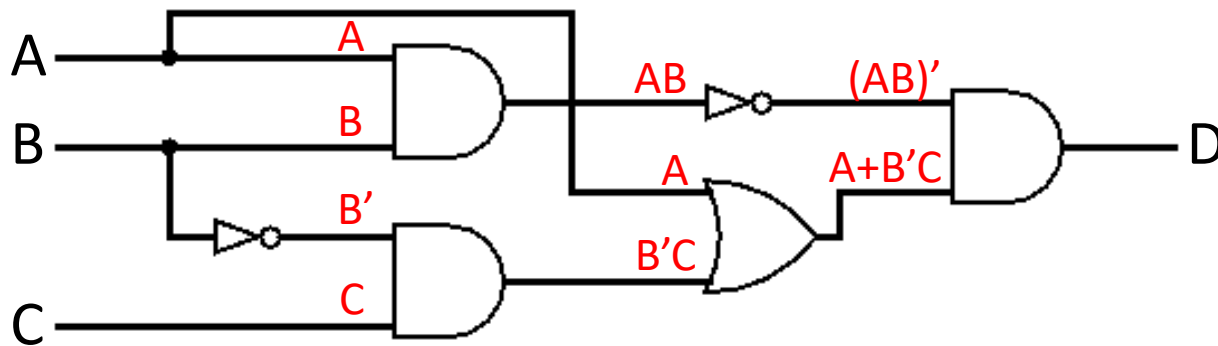
- Simplify the following circuit:



- Options:
  - 1) Test all combinations of the inputs and build the Truth Table, then use SoP or PoS
  - 2) Write out expressions for signals based on gates
    - Will show this method here

# Circuit Simplification Example (2/4)

- Simplify the following circuit:



- Start from left, propagate signals to the right
- Arrive at  $D = (AB)'(A + B'C)$

# Circuit Simplification Example (3/4)

- Simplify Expression:

$$D = (AB)'(A + B'C)$$

$$= (A' + B')(A + B'C)$$

$$= A'A + A'B'C + B'A + B'B'C$$

$$= 0 + A'B'C + B'A + B'B'C$$

$$= A'B'C + B'A + B'C$$

$$= (A' + 1)B'C + AB'$$

$$\left. \begin{aligned} &= B'C + AB' \\ &= B'(A + C) \end{aligned} \right\} \begin{array}{l} \text{Which of these} \\ \text{is "simpler"?} \end{array}$$

DeMorgan's

Distribution

Complementarity

Idempotent Law

Distribution

Law of 1's

Distribution



# Circuit Simplification Example (4/4)

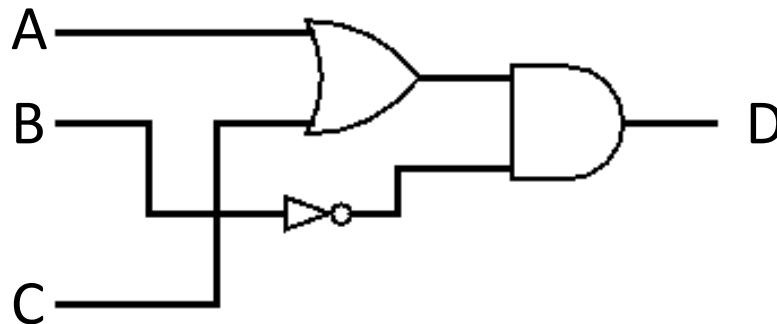
- Draw out final circuit:

$$- D = B'C + AB' = \underbrace{B'(A + C)}_{3}$$

4

How many gates  
do we need for each?

- Simplified Circuit:



– Reduction from 6 gates to 3!

# Karnaugh Maps

- Again, this is completely OPTIONAL material
  - Recommended you use .pptx to view animations
- Karnaugh Maps (K-maps) are an alternate way to simplify Boolean Algebra
  - This technique is normally taught in CS150
  - We will never ask you to use a K-map to solve a problem, but you may find it faster/easier if you choose to learn how to use it
- For more info, see:  
[http://en.wikipedia.org/wiki/Karnaugh\\_map](http://en.wikipedia.org/wiki/Karnaugh_map)

# Underlying Idea

- Using Sum of Products, “neighboring” input combinations simplify
  - “Neighboring”: inputs that differ by a single signal
  - e.g.  $ab + a'b = b$ ,  $a'bc + a'bc' = a'b$ , etc.
  - **Recall:** Each product only appears where there is a 1 in the output column
- **Idea:** Let's write out our Truth Table such that the neighbors become apparent!
  - Need a Karnaugh map for *EACH* output

# Reorganizing the Truth Table

- Split inputs into 2 *evenly-sized* groups
  - One group will have an extra if an odd # of inputs
- Write out all combinations of one group horizontally and all combinations of the other group vertically
  - Group of  $n$  inputs  $\rightarrow 2^n$  combinations
  - Successive combinations change only 1 input

**2 Inputs:**

A \ B	0	1
0		
1		

**3 Inputs:**

C \ AB	00	01	11	10
0				
1				

# K-map: Majority Circuit (1/2)

- Filling in the Karnaugh map:

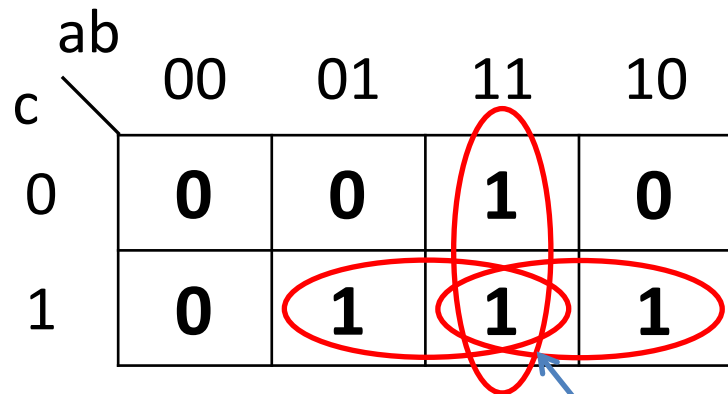
a	b	c	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

c \ ab	00	01	11	10
	0	0	1	0
1	0	1	1	1

- Each row of truth table corresponds to ONE cell of Karnaugh map
- Recommended you view the animation on this slide on the Powerpoint (pptx)
- Note the funny jump when you go from input 011 to 100 (*most mistakes made here*)

# K-map: Majority Circuit (2/2)

- Group neighboring 1's so all are accounted for:
  - Each group of neighbors becomes a product term in output



ab	00	01	11	10
c				
0	0	0	1	0
1	0	1	1	1

Single cell can  
be part of  
many groups

- $y = bc + ab + ac$
- Larger groups become smaller terms
  - The single 1 in top row  $\rightarrow abc'$
  - Vertical group of two 1's  $\rightarrow ab$
  - If entire lower row was 1's  $\rightarrow c$

Q: 为什么某项  
可以被多次重复  
使用?

# General K-map Rules

Q: 为什么最简表达式不唯一?

- Only group in powers of 2
  - Grouping should be of size  $2^i \times 2^j$
  - Applies for both directions
- Wraps around in all directions
  - “Corners” case is extreme example
- Always choose largest groupings possible
  - Avoid single cells whenever possible
- $y = bd + b'd' + acd$

		ab			
		00	01	11	10
cd	00	1	0	0	1
	01	0	1	1	0
	11	0	1	1	1
	10	1	0	0	1

- 1) NOT a valid group
- 2) IS a valid group
- 3) IS a valid group
- 4) “Corners” case
- 5) 1 of 2 good choices here (spot the other?)


# 提纲

- 内容主要取材：CS61C的17讲、18讲
  - <http://inst.eecs.berkeley.edu/~cs61c/su12>
- 晶体管
- 门电路
- 运算
  - 加法、减法、乘法、除法



# Adder/Subtractor: 1-bit LSB Adder

$$\begin{array}{rcccc} & a_3 & a_2 & a_1 & a_0 \\ + & b_3 & b_2 & b_1 & b_0 \\ \hline s_3 & s_2 & s_1 & s_0 \end{array}$$

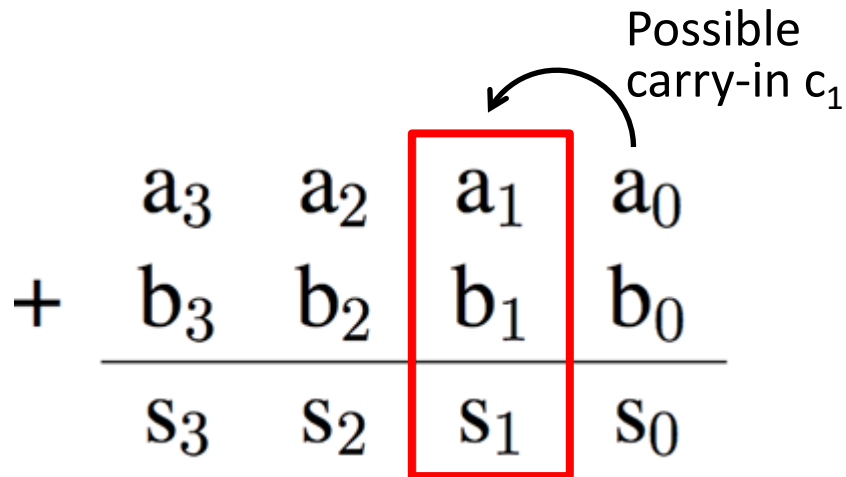
Carry-out bit 

$a_0$	$b_0$	$s_0$	$c_1$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$s_0 = a_0 \text{ XOR } b_0$$

$$c_1 = a_0 \text{ AND } b_0$$

# Adder/Subtractor: 1-bit Adder



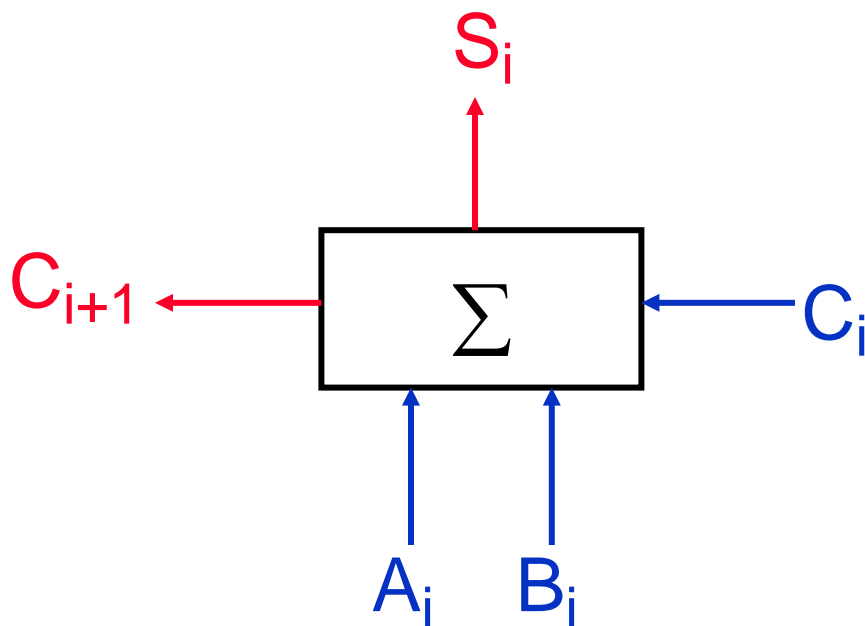
$a_i$	$b_i$	$c_i$	$s_i$	$c_{i+1}$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S_i = \bar{A}_i \bar{B}_i C_i + \bar{A}_i B_i \bar{C}_i + A_i \bar{B}_i \bar{C}_i + A_i B_i C_i$$

$$C_{i+1} = A_i B_i + A_i C_i + B_i C_i$$

## 加减法运算

### ❖ 加法单元（全加器）

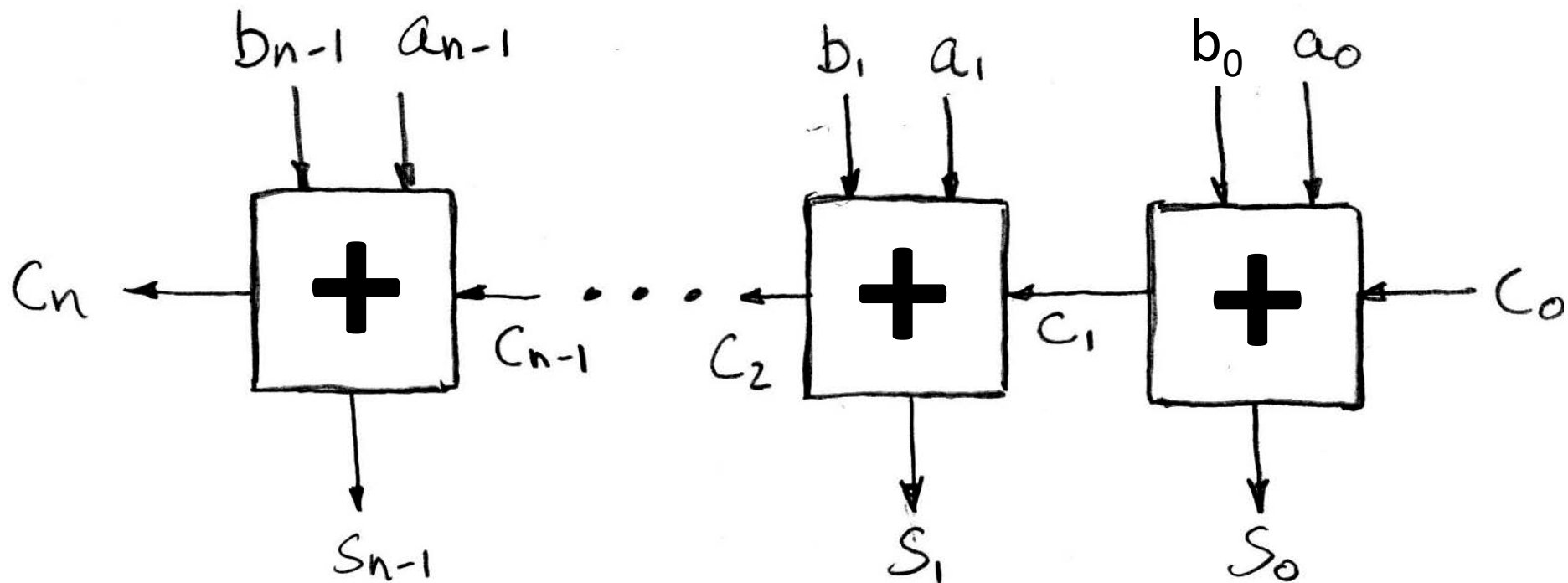


$$S_i = \bar{A}_i \bar{B}_i C_i + \bar{A}_i B_i \bar{C}_i + A_i \bar{B}_i \bar{C}_i + A_i B_i C_i$$

$$C_{i+1} = A_i B_i + A_i C_i + B_i C_i$$

# N x 1-bit Adders $\rightarrow$ N-bit Adder

- Connect  $\text{CarryOut}_{i-1}$  to  $\text{CarryIn}_i$  to chain adders:



# 作业

- 参考书：数字设计和计算机体系结构
- WORD：1.32、1.34、1.48、1.51、1.62、1.63、2.1(e)、2.9(c)、2.16、2.19
  - 2.16、2.19：要求用卡诺图化简
- LogiSim（**不提交**）
  - 构造并模拟1位加法器
  - 构造并模拟4位加法器
    - ◆ 学习使用层次设计
- 作业提交：学校course平台
- 截止时间：10.9的23:59